UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/849,822	05/04/2001	G. Glenn Henry	CNTR: 2050	8898

7590 05/24/2004

James W. Huffman
1832 N. Cascade Ave
Colorado Springs, CO 80907

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	3

DATE MAILED: 05/24/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/849,822	Applicant(s) HENRY ET AL.
	Examiner David J. Huisman	Art Unit 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 May 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 May 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 2.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-39 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: #2. IDS as received on 6/18/2002.

Specification

3. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

4. The disclosure is objected to because of the following informalities: On page 1, all related/copending applications should be identified by serial number or patent number and the attorney docket number should be removed.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claim 18 contains the trademark/trade name “x86”. Where a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of 35 U.S.C. 112, second paragraph. See *Ex parte*

Simpson, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product. A trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus, a trademark or trade name does not identify or describe the goods associated with the trademark or trade name. In the present case, the trademark/trade name is used to identify/describe an instruction set architecture employed by Intel microprocessors and, accordingly, the identification/description is indefinite.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1-7, 27, 33-35, and 38 are rejected under 35 U.S.C. 102(b) as being anticipated by Gochman et al., U.S. Patent No. 5,964,868 (herein referred to as Gochman).

9. Referring to claim 1, Gochman has taught an apparatus in a processor for speculatively performing a return instruction, comprising:

a) first and second call/return stacks, for providing first and second return addresses, respectively. See Fig.1, and column 3, lines 33-53. Note that a speculative (predicted) return address is produced by the speculative stack and an actual return address is produced by the actual return stack.

b) a comparator, coupled to said first and second call/return stacks, for comparing said first and second return addresses, and control logic, coupled to said comparator, for controlling the processor to branch to said first return address, said control logic subsequently controlling the processor to branch to said second return address if said comparator indicates said first and second return addresses do not match. Although a comparator is not explicitly shown in Gochman's figures, this component inherently exists because Fig.3 shows a step in which a determination is made as to whether a branch has been mispredicted or not. In order to determine if a misprediction has been made, the speculative and actual return addresses must be compared. As the Fig.3 shows, if a misprediction did not occur (the speculative return address was correct, i.e., it matches the actual return address), then the process is done and execution continues. However, if a misprediction did occur (the speculative return address was incorrect, i.e., it does not match the actual address), then the instruction fetch must be restarted at the actual return address. See column 7, lines 20-32, for further details.

10. Referring to claim 2, Gochman has taught an apparatus as described in claim 1. Gochman has further taught that said second call/return stack is configured to provide said second return address in response to instruction decode logic decoding a return instruction. See Fig.2 and note that the second (actual) return address is popped off of second (actual) stack during the E (execute) stage. However, it is inherent that execution occurs in response to decoding because a system cannot execute an instruction without knowing the type of instruction. Therefore, since execution occurs in response to decoding, and the popping occurs during execution, then the popping occurs in response to the decoding.

11. Referring to claim 3, Gochman has taught an apparatus as described in claim 1. Gochman has further taught that said first call/return stack speculatively provides said first return address before decoding of said return instruction. See Fig.2 and note that the first (speculative) return address is popped from the first (speculative) stack in the PF stage, which occurs before the D1 and D2 decode stages.

12. Referring to claim 4, Gochman has taught an apparatus as described in claim 3. Gochman has further taught that said first call/return stack speculatively provides said first return address in response to a fetch address, said fetch address selecting a cache line of an instruction cache. See column 6, lines 4-32, and lines 48-56. Note that a fetch address is applied to the branch target buffer cache, which is an instruction cache because it caches information regarding branch instructions.

13. Referring to claim 5, Gochman has taught an apparatus as described in claim 4. Gochman has further taught that said first call/return stack speculatively provides said first return address in response to said fetch address whether or not said return instruction is present in said cache line. See Fig.4 and note that a return address (from stacks 51) is sent to a single multiplexer within branch prediction circuit 40. Therefore, the address is provided regardless of whether or not a return instruction exists. However, the multiplexer will not select the return address prediction if a return instruction does not exist within the cache. For instance, if the cache line includes an ordinary branch instruction (call), but not a return, then the multiplexer will select the address supplied by cache 41 as opposed to the address supplied by stack 51.

14. Referring to claim 6, Gochman has taught an apparatus as described in claim 1. Gochman has further taught a branch target address cache (BTAC), coupled to said first

call/return stack, for caching a plurality of indications of whether a corresponding plurality of instructions previously executed by the processor are return instructions. See Fig.4, component 41, and column 6, lines 4-25, and note that the type of branch, which includes the return-type, is cached.

15. Referring to claim 7, Gochman has taught an apparatus as described in claim 6. Gochman has further taught that said first call/return stack provides said first return address in response to said BTAC providing one of said plurality of indications, wherein said one of said plurality of indications indicates that said corresponding instruction is a return instruction. See column 6, lines 4-56. It should be noted that when a "hit" occurs in the cache, the system has fetched a branch instruction. The cache determines what type of branch it is (call, return, etc.) and based on the type, the appropriate action is performed. In the case of a return instruction, the speculative stack is popped.

16. Referring to claim 27, Gochman has taught a method for speculatively branching a microprocessor to a target address of a return instruction, comprising:

a) generating a first target address by a first call/return stack. See Fig.1, component 51 (speculative return stack), and note that it generates a return target address in the PF stage shown in Fig.2

b) branching to said first target address. See column 4, lines 35-45.

c) generating a second target address by a second call/return stack subsequent to said branching to said first target address. See Fig.1, component 55 (actual return stack), and note that it generates a return target address in the E stage shown in Fig.2.

d) comparing said first and second target addresses. See column 7, lines 20-22.

e) and branching to said second target address if said first and second target addresses do not match. See column 7, lines 26-30, and Fig.3 (the "Branch Mispredicted?" step). Note that when the addresses are compared, if they are different, then the return was mispredicted. This causes the fetch unit to restart fetching at the second (actual) target address.

17. Referring to claim 33, Gochman has taught a method as described in claim 27. Gochman has further taught that said generating said first target address comprises popping said first target address off said first call/return stack. See column 6, lines 48-56. Note that after the return address is obtained, the top-of-stack pointer (TOS pointer 53) is incremented, effectively popping the stack.

18. Referring to claim 34, Gochman has taught a method as described in claim 33. Gochman has further taught pushing said first target address onto said first call/return stack prior to said popping said first target address off said first call/return stack. See column 2, lines 29-37.

19. Referring to claim 35, Gochman has taught a method as described in claim 34. Gochman has further taught calculating said first target address prior to said pushing. See column 4, lines 30-33, and note that the address that is pushed onto the stack is the address of the instruction following the call instruction. Therefore, that address is calculated with respect to the call instruction and then pushed onto the stack. See column 6, lines 44-47, as well, for the calculation means.

20. Referring to claim 38, Gochman has taught a method as described in claim 34. Gochman has further taught that said pushing is performed in response to an instruction cache fetch address. See column 6, lines 4-47, and note that in response to a fetch address, if a call is detected, a push will be performed.

Claim Rejections - 35 USC § 103

21. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

22. Claims 8, 14-26, 28-29, 31-32, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gochman, as applied above.

23. Referring to claim 8, Gochman has taught an apparatus as described in claim 7. Gochman has not explicitly taught that said BTAC provides said one of said plurality of indications in response to an instruction cache fetch address. However, Official Notice is taken that instruction caches from which instructions are fetched are well known and accepted in the art. Instructions are always fetched from some form of memory, whether it be an instruction cache, main memory, etc. However, an instruction cache is much faster than main memory, and therefore, to increase fetching speed, and ultimately execution speed, it would have been obvious to one of ordinary skill in the art at the time of the invention to have Gochman's instruction fetch unit (Fig.4, component 30) coupled to an instruction cache.

24. Referring to claim 14, Gochman has taught a microprocessor for predicting return instruction target addresses.

a) Gochman has not explicitly taught an instruction cache, for providing a line of instruction bytes selected by a fetch address, said fetch address provided on an address bus. However, Official Notice is taken that instruction caches from which instruction bytes are fetched are well known and accepted in the art. Instructions are always fetched from some form of memory, whether it be an instruction cache, main memory, etc. However, an instruction cache is much faster than main memory, and therefore, to increase fetching speed, and ultimately execution speed, it would have been obvious to one of ordinary skill in the art at the time of the invention to have Gochman's instruction fetch unit (Fig.4, component 30) coupled to an instruction cache. Also, addresses are inherently placed on an address bus.

b) Gochman has taught further a speculative branch target address cache (BTAC), coupled to said address bus, for caching indications of previously executed return instructions, said speculative BTAC providing one of said indications to a speculative call/return stack in response to said fetch address whether or not a return instruction is present in said line of instruction bytes. See Fig.4, component 41, and column 6, lines 4-56, and note that the type of branch, which includes the return-type, is cached. It should be noted that when a "hit" occurs in the cache, the system has fetched a branch instruction. The cache determines what type of branch it is (call, return, etc.) and based on the type, the appropriate action is performed. In the case of a return instruction, the speculative stack is popped.

c) said speculative call/return stack coupled to said speculative BTAC, for providing a speculative return address to address selection logic in response to said one of said indications indicating one of said previously executed return instructions is potentially present in said line of instruction bytes. See column 6, lines 4-32, and lines 48-56, and Fig.4, component 51. If a return instruction is encountered, then the stack will be popped. Note that a return instruction is potentially present in the cache line because the cache line has the capability of holding a return instruction.

d) said address selection logic configured to select said speculative return address as a subsequent fetch address for provision to said instruction cache. See column 4, lines 35-45, and column 6, lines 48-56. Note that when a return address is predicted, the address is used to fetch instructions without delay.

25. Referring to claim 15, Gochman has taught a microprocessor as described in claim 14. Gochman has further taught address generation logic, coupled to said speculative call/return stack, for calculating said speculative return address for pushing onto said speculative call/return stack. See Fig.4 and note the "32-BIT INC" component. Also, see column 6, lines 44-47.

26. Referring to claim 16, Gochman has taught a microprocessor as described in claim 16. Gochman has further taught that said speculative BTAC is configured to cache indications of previously executed call instructions, said speculative BTAC providing one of said indications of one of said previously executed call instructions in response to said fetch address. See column 6, lines 4-32, and note that in response to a fetch address, the cache will provide information regarding the type of branch instruction (indication), which is used in making a prediction from an appropriate source (stack or cache).

27. Referring to claim 17, Gochman has taught a microprocessor as described in claim 16. Gochman has further taught that said address generation logic calculates said speculative return address in response to said branch target address cache providing said one of said indications of one of said previously executed call instructions. See column 6, lines 4-47.

28. Referring to claim 18, Gochman has taught a microprocessor as described in claim 14. Furthermore, it is inherent that previously executed return instruction are x86 RET instructions

because the x86 instruction set is an Intel instruction set and Gochman's system is an Intel system (note the assignee of the front page).

29. Referring to claim 19, Gochman has taught a microprocessor for predicting return instruction target addresses.

a) Gochman has not taught an instruction cache, for generating a line of instruction bytes selected by a fetch address, said fetch address received from an address bus. However, Official Notice is taken that instruction caches from which instruction bytes are fetched are well known and accepted in the art. Instructions are always fetched from some form of memory, whether it be an instruction cache, main memory, etc. However, an instruction cache is much faster than main memory, and therefore, to increase fetching speed, and ultimately execution speed, it would have been obvious to one of ordinary skill in the art at the time of the invention to have Gochman's instruction fetch unit (Fig.4, component 30) coupled to an instruction cache. Also, addresses are inherently placed on an address bus.

b) address selection logic, coupled to said address bus, for selecting said fetch address and providing said fetch address on said address bus. See Fig.4, and note the multiplexer which selects an address from either the stack(s), the branch target cache, or the write-results stage of the pipeline. This logic will determine the next fetch address which will be applied on the address bus by fetch logic 30.

c) a branch target address cache (BTAC), coupled to said address bus, for caching indications of previously executed return instructions and for providing one of said indications in response to said fetch address. See Fig.4, component 41, and column 6, lines 4-56, and note that the type of branch (indication), which includes the return-type, is cached. It should be noted that when a "hit" occurs in the cache (in response to an applied fetch address), the system has fetched a branch instruction. The cache determines what type of branch it is (call, return, etc.) and based on the type, the appropriate action is performed. In the case of a return instruction, the speculative stack is popped.

d) a first call/return stack, coupled to said BTAC, for providing a first return address to said address selection logic in response to said one of said indications. See column 6, lines 4-56, and Figs.1 and 4, component 51.

e) decode logic, coupled to said instruction cache, for decoding said line of instruction bytes. See Fig.4, component 60.

f) a second call/return stack, coupled to said decode logic, for providing a second return address to said address selection logic in response to said decode logic indicating that a return instruction is present in said line of instruction bytes. See Figs.1 and 4, component 55. Note that this stack provides an actual address, via popping, during the E (execute) stage (Fig.2). It is inherent that execution occurs in response to decoding because a system cannot execute an instruction without knowing the type of instruction. Therefore, since execution occurs in response to decoding, and the popping occurs during execution, then the popping occurs in response to the decoding.

30. Referring to claim 20, Gochman has taught a microprocessor as described in claim 19. Gochman has further taught that said first call/return stack provides said first return address before said decode logic decodes said line of instruction bytes. See Fig.2 and note that the first (speculative) return address is popped from the first (speculative) stack in the PF stage, which occurs before the D1 and D2 decode stages.

31. Referring to claim 21, Gochman has taught a microprocessor as described in claim 19. Gochman has further taught that said branch target address cache provides said one of said indications in response to said fetch address whether or not a return instruction is present in said line of instruction bytes. For instance, if a call instruction is encountered, then the branch target address cache will provide an indication indicating that the encountered instruction is a call instruction. More specifically, the cache line does not have to include a return instruction for an indication to be provided by the branch cache.

32. Referring to claim 22, Gochman has taught a microprocessor as described in claim 19. Gochman has further taught that said first call/return stack provides said first return address in response to said one of said indications indicating said one of said previously executed return instructions is potentially present in said line of instruction bytes. See column 6, lines 4-56. It should be noted that when a "hit" occurs in the cache, the system has fetched a branch instruction. The cache determines what type of branch it is (call, return, etc.) and based on the type, the appropriate action is performed. In the case of a return instruction, the speculative stack is popped. Note that a return instruction is potentially present in the cache line because the cache line has the capability of holding a return instruction.

33. Referring to claim 23, Gochman has taught a microprocessor as described in claim 19. Gochman has further taught control logic, coupled to said BTAC, configured to control said address selection logic to select said first return address during a first period. Note that the first address is supplied during the PF stage shown in Fig.2. It is inherent that some control logic exists which makes this address available at this particular stage.

34. Referring to claim 24, Gochman has taught a microprocessor as described in claim 23. Gochman has further taught a comparator, coupled to said first and second call/return stacks, for comparing said first and second return addresses. Although a comparator is not explicitly shown in Gochman's figures, this component inherently exists because Fig.3 shows a step in which a determination is made as to whether a branch has been mispredicted or not. In order to determine if a misprediction has been made, the speculative and actual return addresses must be compared.

35. Referring to claim 25, Gochman has taught a microprocessor as described in claim 24. Gochman has further taught that said control logic is further configured to control said address selection logic to select said second return address subsequent to controlling said address selection logic to select said first return address if said comparator indicates said first and second return addresses do not match. Note that if the second address does not match the first, a misprediction has occurred, and the system must being fetching at the second (actual) address which is provided during the execution (E) stage, which is after the stage in which the first (speculative) address was selected. See Fig.2.

36. Referring to claim 26, Gochman has taught a microprocessor as described in claim 19. Gochman has further taught that said second call/return stack provides said second return address subsequent to said first call/return stack providing said first return address. See Fig.2 and note that the speculative return address is supplied in the PF stage and the actual return address is supplied during the E stage.

37. Referring to claim 28, Gochman has taught a method as described in claim 27. Gochman has not explicitly taught that said branching to said first target address comprises selecting said

first target address and providing said first target address as a fetch address to an instruction cache in the microprocessor. However, Official Notice is taken that instruction caches from which instruction bytes are fetched are well known and accepted in the art. Instructions are always fetched from some form of memory, whether it be an instruction cache, main memory, etc. However, an instruction cache is much faster than main memory, and therefore, to increase fetching speed, and ultimately execution speed, it would have been obvious to one of ordinary skill in the art at the time of the invention to have Gochman's instruction fetch unit (Fig.4, component 30) coupled to an instruction cache. Also, addresses are inherently placed on an address bus. Finally, it should be realized from column 6, lines 51-54, and column 4, lines 41-45, that the first address is applied to an instruction cache in order to begin fetching instructions from the predicted path.

38. Referring to claim 29, Gochman has taught a method as described in claim 28. Gochman has further taught that said generating said first target address comprises said first call/return stack generating said first target address in response to a previous fetch address that was provided to said instruction cache. From the basic description given in column 4, lines 4-56, a fetch address is applied to the branch target address cache. This fetch address also corresponds to a particular instruction which may or may not be a branch (call, return, etc). If a hit occurs in the cache, then the instruction is a branch, and based on another field in the cache, if that branch is a return instruction, then the selected return address will be the one popped from the top of the first stack.

39. Referring to claim 30, Gochman has taught a method as described in claim 29. Gochman has further taught that said generating said first target address is performed whether or not a

return instruction is present in an instruction cache line selected by said fetch address. See Fig.4 and note that a return address (from stacks 51) is sent to a single multiplexer within branch prediction circuit 40. Therefore, the address is provided regardless of whether or not a return instruction exists. However, the multiplexer will not select the return address prediction if a return instruction does not exist within the cache; that is, it is made available, but it is not necessarily utilized. For instance, if the cache line includes an ordinary branch instruction (call), but not a return, then the multiplexer will select the address supplied by cache 41 as opposed to the address supplied by stack 51.

40. Referring to claim 31, Gochman has taught a method as described in claim 29. Gochman has further taught decoding a return instruction present in a line of instruction bytes selected from said instruction cache by said fetch address, wherein said decoding said return instruction present in said line of instruction bytes is performed subsequent to said branching to said first target address. Note from Fig.2 and column 4, lines 35-45, that the first address is obtained in the PF pipeline stage. By doing this, fetching instructions is continued so no stalling. The return instruction itself won't be fully decoded until three cycles later in the D2 stage.

41. Referring to claim 32, Gochman has taught a method as described in claim 31. Gochman has further taught that said generating said second target address comprises said second call/return stack generating said second target address in response to said decoding said return instruction present in said line of instruction bytes. See Fig.2 and note that the second (actual) return address is popped off of second (actual) stack during the E (execute) stage. However, it is inherent that execution occurs in response to decoding because a system cannot execute an instruction without knowing the type of instruction. Therefore, since execution occurs in

response to decoding, and the popping occurs during execution, then the popping occurs in response to the decoding.

42. Referring to claim 39, Gochman has taught a microprocessor for predicting return instruction target addresses.

a) Gochman has not explicitly taught an instruction cache, for providing a line of instructions in response to a fetch address received on an address bus. However, Official Notice is taken that instruction caches from which instruction bytes are fetched are well known and accepted in the art. Instructions are always fetched from some form of memory, whether it be an instruction cache, main memory, etc. However, an instruction cache is much faster than main memory, and therefore, to increase fetching speed, and ultimately execution speed, it would have been obvious to one of ordinary skill in the art at the time of the invention to have Gochman's instruction fetch unit (Fig.4, component 30) coupled to an instruction cache. Also, addresses are inherently placed on an address bus.

b) a multiplexer, having a plurality of inputs, configured to select one of said plurality of inputs for provision on said address bus as said fetch address to said instruction cache. See Fig.4, and note the multiplexer which receives input from the branch target buffer cache, one of the two stacks, and the write-results stage of the pipeline. This multiplexer will select one of multiple addresses that will be sent on the address bus to the instruction cache.

c) a speculative branch target address cache (BTAC), coupled to said address bus, for indicating a speculative presence of a return instruction in said line of instructions. See Fig.4, component 41, and column 4, lines 16-25.

d) a speculative call/return stack, coupled to said speculative BTAC, for providing a speculative return address to a first of said plurality of multiplexer inputs in response to said speculative BTAC indicating said speculative presence of said return instruction. See Fig.1 and Fig.4, component 51, and column 6, lines 4-25, and lines 48-56.

e) decode logic, configured to receive and decode said line of instructions. See Fig.4, component 60.

f) a non-speculative call/return stack, coupled to said decode logic, for providing a non-speculative return address to a second of said plurality of multiplexer inputs in response to said decode logic indicating that said return instruction is actually present in said line of instructions. See Fig.1 and Fig.4, component 55. Note that this stack provides an actual (non-speculative) address, via popping, during the E (execute) stage (Fig.2). It is inherent that execution occurs in response to decoding because a system cannot execute an instruction without knowing the type of instruction. Therefore, since execution occurs in response to decoding, and the popping occurs during execution, then the popping occurs in response to the decoding.

g) a comparator, coupled to said speculative and non-speculative call/return stacks, for comparing said speculative and non-speculative return addresses. Although a comparator is not explicitly shown in Gochman's figures, this component inherently exists because Fig.3 shows a step in which a determination is made as to whether a branch has been mispredicted or not. In order to determine if a misprediction has been made, the speculative and actual return addresses must be compared. See column 7, lines 20-22.

h) wherein said multiplexer selects said speculative return address in a first instance, and selects said non-speculative return address in a second instance subsequent to said first instance if said comparator indicates that said speculative and non-speculative return addresses do not match. As the Fig.3 shows, if a misprediction did not occur (the speculative return address was correct, i.e., it matches the actual return address), then the process is done and execution continues. However, if a misprediction did occur (the speculative return address was incorrect, i.e., it does not match the actual address), then the instruction fetch must be restarted at the actual return address. Also, see column 7, lines 20-32, and Fig.2 for further details.

43. Claims 9-13 and 36-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gochman, as applied above, in view of Hilgendorf et al., U.S. Patent No. 5,974,543 (herein referred to as Hilgendorf).

44. Referring to claim 9, Gochman has taught an apparatus as described in claim 6. Gochman has not taught that said BTAC is further configured to cache a plurality of lengths of a corresponding plurality of call instructions previously executed by the processor. However, Hilgendorf has taught such a concept. See column 8, line 60, to column 9, line 16, and note that the BHT (branch history table / BTAC), provides the length of the call instruction so that it can be added to the fetch address (i.e., the address of the call instruction) to realize the return address. A person of ordinary skill in the art would have recognized that such a system gives the user increased flexibility in that return addresses can be calculated in variable instruction length systems, where instruction length is not always constant. Consequently, in order to achieve this flexibility, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify the BTAC of Gochman to include the length of the branch instruction.

45. Referring to claim 10, Gochman in view of Hilgendorf has taught an apparatus as described in claim 9. Hilgendorf has further taught that said first return address comprises a sum of an instruction cache fetch address and one of said plurality of lengths provided by said BTAC. Again, see column 8, line 64, to column 9, line 5.

46. Referring to claim 11, Gochman in view of Hilgendorf has taught an apparatus as described in claim 10. Hilgendorf has further taught that said BTAC is further configured to cache a plurality of byte offsets within an instruction cache line of said corresponding plurality of call instructions, said byte offsets being within an instruction cache line selected by said fetch address. See column 5, lines 20-24. More specifically, recall that Hilgendorf has taught caching the instruction length. Hilgendorf has further taught caching byte offsets used for addressing an instruction cache, which is disclosed in column 3, lines 24-35.

47. Referring to claim 12, Gochman in view of Hilgendorf has taught an apparatus as described in claim 11. Furthermore, it is inherent that said instruction cache line is selected by said fetch address. When you fetch a line from cache, you want to fetch the line corresponding to the fetch address.

48. Referring to claim 13, Gochman in view of Hilgendorf has taught an apparatus as described in claim 10. Hilgendorf has further taught that said first return address comprises a sum of said instruction cache fetch address and said one of said plurality of lengths and one of said plurality of byte offsets. See column 9, lines 17-20.

49. Referring to claim 36, Gochman has taught a method as described in claim 35. Gochman has not taught that said calculating said first target address comprises adding a cached length of a previously cached call instruction and a fetch address selecting an instruction cache line

potentially including said previously executed call instruction. However, Hilgendorf has taught such a concept. See column 8, line 60, to column 9, line 16, and note that the BHT (branch history table / BTAC), provides the length of the call instruction so that it can be added to the fetch address (i.e., the address of the call instruction) to realize the return address. A person of ordinary skill in the art would have recognized that such a system gives the user increased flexibility in that return addresses can be calculated in variable instruction length systems, where instruction length is not always constant. Consequently, in order to achieve this flexibility, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify the BTAC of Gochman to include the length of the branch instruction.

50. Referring to claim 37, Gochman in view of Hilgendorf has taught a method as described in claim 36. Hilgendorf has further taught that said generating said first target address comprises adding said fetch address, said cached length, and a cached offset of said call instruction within said instruction cache line. See column 9, lines 17-20.

Conclusion

51. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

Sakamoto et al., "Microarchitecture Support for Reducing Branch Penalty in a

Superscalar Processor," 1996, pp.208-216, has taught dual return stacks for return address prediction.

D'Sa et al., U.S. Patent No. 6,151,671, has taught a system and method of maintaining and utilizing multiple return stack buffers.

McDonald, U.S. Patent No. 6,314,514, has taught a method and apparatus for correcting an internal call/return stack in a microprocessor that speculatively executes call and return instructions.

Black et al., U.S. Patent No. 5,761,723, has taught a data processor with branch prediction that will provide a branch prediction corresponding to a fetch address even if a branch instruction is not located at the fetch address due to a context switch.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
May 14, 2004

[illegible]

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100